DEPARTMENT OF THE AIR FORCE

HEADQUARTERS ELECTRONIC SYSTEMS CENTER (AFMC)
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731-5000

JAN 3 0 1996

MEMORANDUM FOR LORAL FEDERAL SYSTEMS
ATTN: RICHARD HUGHES

FROM: ESC/ENS
5 Eglin Street
Bldg. 1704, Rm. 206
Hanscom AFB, MA 01731-2116

SUBJECT: Upgrade to Distribution Statement A

1. The STARS product, CDRL A014-014, "Front-End Process Definition for Projects Engaged in Significant Technology Transition", is upgraded to Distribution Statement A effective 29 Jan 96.

2. Please direct any questions you may have to the Jim Henslee at (617) 377-8563.

ROBERT LENCEWICZ
ESC CARDS Program Manager
Software Design Center
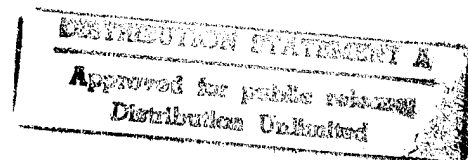
DISTRIBUTION STATEMENT UPGRADE

CDRL Number and Task Number: CDRL A014-014, Task IV02

Product Title and Brief Description (what it is and what it does): "Front-End Process Definition for Projects Engaged in Significant Technology Transition"

This paper provides experience-based guidelines to help organizations deal with the issues associated with front-end process definition.
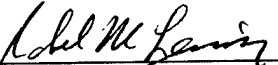
Date Delivered to the Program Office: 24 Jan 96

Reviewer's Name, Extension Number and Date of Review: Marcelle Nachef, (617) 377-4918, 29 Jan 96

Intended Audience: Public conferences, trade shows, workshops, and
STC '96

Comments:

The STARS product, Front-End Process Definition for Projects Engaged in Significant Technology Transition, previously under Distribution Statement C, is upgraded to Distribution Statement A effective 29 Jan 96. This product is generic and does not apply to specific defense articles and defense services. In accordance with Memorandum of Agreement between ESC/PA and ESC/ENS concerning upgrades of STARS products to Distribution A, the STARS program office at ESC/ENS has reviewed this product and has determined that the information is unclassified, technically accurate, and suitable for public release.

_____
Approving Authority

_____
1/30/96
Date

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

Technical Papers:
Front-End Process Definition for Projects Engaged In Significant Technology Transition

Contract No. F19628-93-C-0129

Task IV01 - Megaprogramming Transition Support

Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731-2116

Prepared by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

19960611 174

CLEARED FOR PUBLIC RELEASE, DISTRIBUTION IS UNLIMITED

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

Technical Papers:
Front-End Process Definition for Projects Engaged In Significant Technology Transition

Contract No. F19628-93-C-0129

Task IV01 - Megaprogramming Transition Support

Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731-2116

Prepared by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

REPORT DOCUMENTION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1/23/96	Initial

4. TITLE AND SUBTITLE

Front-End Process Definition for Projects Engaged in Significant Technology Transition

5. FUNDING NUMBERS

F19628-93-C-0129

6. AUTHOR(S)

Dr. Richard L. Randall/Loral Federal Systems
Richard Drake/Loral Federal Systems

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD   20879

8. PERFORMING ORGANIZATION REPORT NUMBER

A014-014

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Electronic Systems Center/ENS
Air Force Materiel Command, USAF
5 Eglin Street, Building 1704
Hanscom Air Force Base, MA   01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

N/A

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Cleared for Public Release, Disribution is Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Much has been written about process definition, but little of it deals with its very early stages - starting with formulating a basic approach. With severe pressure from all fronts to reengineer ways of doing business, many organizations are trying to cope with just this problem as they wrestle with transitioning to new technologies. In the presence of borderline chaos, an organization's impreative to achieve a CMM Level 3 rating can seem hopelessly remote.

For the last few years, the authors have been involved with the Air Force/STARS Demonstration Project - a project charged with just such an overhaul: namely, to establish a product-line process for a domain of large C2 applications. The team has experienced the gamut from chaos to well-defined processes and has learned many lessons along the way. As the Demonstration Project comes to a successful close and the Air Force continues with the transition, some preocess areas are still evolving. In fact, we now recognize this as a normal, steady state condition - as most organizations in today's rapidly changing world will acknowledge.

This paper provides experience-based guidelines to help organizations deal with the issues associated with front-end process definition.

14. SUBJECT TERMS

Process Definition, Process Architecture, Technology Transition

15. NUMBER OF PAGES

29

16. PRICE CODE

N/A

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

Front-End Process Definition for Projects Engaged in Significant Technology Transition

1.0 Introduction

Much has been written about process definition, but little of it deals with its very early stages, while the organization is overhauling its basic approach (or even defining a brand-new one). With severe pressure from all fronts to reengineer ways of doing business, many organizations are trying to cope with just this problem as they wrestle with transitioning to new technologies. In the presence of borderline chaos, an organization's imperative to achieve a CMM Level 3 rating can seem hopelessly remote.

For the last few years, the authors have been involved with the Air Force/STARS Demonstration Project - a project charged with just such an overhaul: namely, to establish a product-line process for a domain of large C^2 applications. The team has experienced the gamut from chaos to well-defined processes and has learned many lessons along the way. As the Demonstration Project comes to a successful close and the Air Force continues with the transition, some process areas are still evolving. In fact, we now recognize this as a normal, steady state condition - as most organizations in today's rapidly changing world will acknowledge.

This paper provides experience-based guidelines to help organizations deal with the issues associated with front-end process definition. The guidelines are couched in the guise of a high-level *approach definition process*, consisting of the following steps, which are executed iteratively as the approach matures:

- Analyze/update the process requirements
- Assess the organization's posture
- Plan first/next architecture refinement increment
- Define/update the Process Architecture, by creating/refining "process element groups"

Fundamental to these guidelines is the notion of *Process Architecture*, which we believe is instrumental in managing approach definition. In fact, it is useful to think of "Approach Definition" as equating to "Process Architecture Definition". Accordingly, as part of the guidelines, we provide a working definition of a Process Architecture and offer some criteria for judging its quality.

1.1 Context

The authors' experience upon which this paper is based was gained from involvement in many systems development activities - notably the recently completed Air Force/STARS (Software Technology for Adaptable, Reliable Systems) Demonstration Project.

This Demonstration Project was the result of a partnership between the Advanced Research Projects Agency (ARPA) STARS program and the Space and Warning Systems Center (SWSC), now part of Air Force Material Command, located at Peterson AFB. The SWSC supports Command and Control Systems for North American Aerospace Defense Command (NORAD), United States Space Command (USSPACECOM), and Air Force Space Command (AFSPC). Currently this involves responsibility for 34 systems, comprising over 12 million lines of code developed in 27 different languages on a variety of hardware platforms.

In the late 80's, the SWSC undertook a new initiative to create an architecture-based approach to solving their desire to provide new capability to the warfighter faster, better, and cheaper. As a result of these initiatives, the SWSC was selected by HQ Air Force to form a partnership with the STARS program and was supported by Loral FS, a STARS prime contractor. The Demonstration Project was undertaken to further the architecture by applying a product-line approach to the development and evolution of software systems. The three-year project resulted in the initial creation of a software product-line which has produced two command and control systems, built in support of Cheyenne Mountain missions.

One of the major challenges of the Demonstration Project was managing the transition of a significant number of new technologies into a group consisting of a heterogeneous mix of Air Force, contractor, and support personnel, with the

objective of emerging with a credible new product-line approach. Technological change was targeted along three major fronts, as shown in Table 1.

Technological Area	Significant New Aspects Targeted
Architecture-Based Reuse	• TAFIM-compliant architectural infrastructure • Two-Lifecycle software engineering process model (domain/application engineering) • Object-oriented design and analysis
Systematic Process	• Sustainable process definition process; and defined processes for key software processes • Cleanroom software engineering process (rigorous, team-oriented, specification-based, statistically certified) • Ada Process Model (incremental, demonstration-based development) • Process enactment, with automated support in selected areas
Automated Support (SEE)	• Code generation via 4GL tools geared to the architectural infrastructure • OO modeling support • Integrated Ada development support • Process modeling support; process enactment support

Table 1:

Readers interested in these specific technologies and the way in which they were adapted/integrated should refer to the AF/STARS Demonstration Project Experience Report [DemoExp96], available in the second quarter of 1996, and accessible through the Loral/STARS WWW pages: **http://source.asset.com/stars/loral/home.html**.

1.2 Overview of Paper

As might be surmised, the journey from the initial state of this project to a credible starting point for a product-line process has **not** been easy. All of the above technologies were successfully assimilated to one degree or another on the project - none perfectly, and some with a good deal of re-interpretation and adjustment. Others, not mentioned, were considered and discarded along the way. In retrospect, most participants feel:

- the project's technology transition goals were overly ambitious, and
- we could have better managed the buildup of our approach.

The purpose of this paper is not to elaborate any of the specifics of this project, which have been amply covered in several other references [e.g., DemoExp96], but rather to communicate some of what we've learned about the "Approach Definition Process". The remainder of the paper is organized as follows:

- **Section 2.0, Lessons Learned** - cites some of the relevant experience that the Demonstration Project organization would take into account if it were starting over - i.e., improvement opportunities for the Approach Definition Process
- **Section 3.0, Approach Definition Guidelines** - offers some suggestions about how an organization with similarly ambitious Technology Transition (TT) objectives might manage the build-up of their own new approach.
- **Section 4.0, Summary** - recaps the major points of the paper.

While reading the paper, please bear two points in mind:

- *The paper focuses on the special problems faced by organizations attempting **revolutionary** change, as distinguished from **evolutionary** process improvement.*

- *We are offering relevant experience and recommended guidelines based on that experience; we are **not** offering a prescriptive solution.*

2.0 Lessons Learned

The Demonstration Project placed a heavy emphasis on process technology, targeting objectives in process definition, process driven planning, process enactment, and measurement - as well as automated support in all of these areas. In retrospect, although the project's accomplishments fell short of expectations, the organization did succeed in transforming itself into a very process-aware and, in some areas, a process-driven organization.

Having wrestled with such a wide range of objectives - with the support of process expertise from STARS, the SEI, and the STSC - the project's experience provides a rich volume of experience. The Air Force/STARS Demonstration Project Experience Report [DemoExp96] provides a comprehensive recount of this experience.

The Experience Report lessons included in this section are those that seem most relevant to the issues associated with this paper's focus: front-end approach definition[1].

2.1 Technology Transition in General

Lesson: Transition to a product-line approach must be:

- *structured - based on organizational goals/priorities and backed by a phased plan*
- *user-sponsored - process definers/owners must be the users of the processes*
- *incremental - define to the detail commensurate with process maturity; pilot before committing*
- *iterative - recognize that maturity only comes through improvement and evolution*

A managed build-up of the approach will guard against premature adoption, which can adversely impact both productivity and morale.

This may be the single most important lesson of the entire Demonstration Project, and its relevance to this paper's topic is clear. The guidelines presented in Section 3.0 will stress a user-sponsored, incremental, iterative approach.

Lesson: Architecture is the foundation of product-line reuse: think architecture first, then process, then SEE - and then plan to iterate on all of them simultaneously.

This is also a fundamental Demonstration Project lesson: if you are trying to establish a product-line (a primary goal of the Demonstration Project) the product-line's unifying software architecture represents a central requirement to be addressed by the organization's process. Although this paper is meant to apply to a wider scope than just product-line organizations, it also seems clear to the authors that having a unifying software architecture greatly enhances the opportunity for process reuse across an organization.

2.2 Process Representation

For the Demonstration Project, a technology group under the leadership of an Air Force officer was in charge of process engineering - responsible for evolving the "process engineering process" and working with practitioners in the various disciplines to help them define their technical, management, and administrative processes. Here is a subset of their lessons:

Lesson: Process descriptions need to meet the needs of the customer (the process user).

This lesson emphasizes that the single most important customer of a process engineering group is the practitioner: the people who will follow - and own - the processes being defined. While the discipline of process engineering may ultimately be served by model rigor and completeness, there is a strong need to "keep it simple" - at least during early phases of process evolution. Users grappling with basic approach issues don't need the added burden of dealing with abstruse representations.

Lesson: $IDEF_0$ is easily misunderstood by process enactors.

Lesson: Process technologies should be combined to meet the differing needs or objectives of an organization. A single technology does not capture all the process information required.

---

[1]. It should be noted that the authors have exercised editorial license in selecting the subset of the project's lessons learned and in grouping them in ways that help communicate points of particular relevance to this paper. In addition, in some cases we have added our own interpretations or elaborations. Interested readers are urged to refer to the original Experience Report for the project's official statements of lessons learned.

These Experience Report lessons are cited back-to-back to illustrate that no one notation suffices for all purposes - or for all audiences. In fact, the Demonstration Project has explored a number of representation techniques, including the following:

- $IDEF_0$ (activity model with information flow),
- $IDEF_{1X}$ (conceptual information model),
- ETVX (Entry/Task/Validation/eXit - for activity sequencing),
- Process Definition Information Organizer Templates [PhillipsEtt95] (for enactment information),
- State Transition Diagrams (for management-level views of process milestones), and
- Ad hoc diagrams and text.

We have attached Appendix A, List of Selected Process Representation Notations, to enumerate and define several candidate process representation notations - including references to texts and articles that provide detailed treatments.

2.3 Automated Support for Modeling

Lesson: Maintaining detailed process information is unwieldy without integrated automated support.

Lesson: Maintaining process information in multiple forms poses a significant maintenance challenge.

The above two Experience Report lessons are actually closely related to the prior two lessons (in which it was asserted that multiple notations are necessary for reflecting process information). With multiple notations - and particularly when process information becomes detailed and voluminous - automated tool support is called for if there is any hope to maintain the information. Further, to avoid the need to record (and maintain!) redundant information, it is highly desirable to have an integrated toolset which shares a single database of process information. The project found that no one toolset fits the bill.

2.4 Incremental Buildup

Lesson: Do not underestimate the break-in time for significantly new processes.

Lesson: In conjunction with the incremental buildup of the organization's process and its development of supporting process technologies, pilot activities at key process transition points are essential.

A bit of context is useful to better understand these two lessons. The Demonstration Project had three major project goals:

- build a real system,
- build up a product-line process, and
- initiate the institutionalization of that process throughout the parent organization.

Even with a very capable team, and even with a fair amount of background in the domain and in many of the technologies, the project found itself trying to learn from experience on many interrelated fronts simultaneously. The engineering team was trying to learn the domain for the software system, several new methods, and several new tools - all at the same time. Added to this was the requirement to synthesize a process, define it, and enact it - even while the organization was struggling with how to do process engineering in the first place! To reach a successful conclusion to the project, the team had to cut back on its ambitions in some areas - including its process ambitions. Along the way, they learned two key truths, which are reflected in the above lessons:

- Technical understanding must be built up incrementally, via a Plan/Do/Learn spiral model [CFRP95]. Further, the challenge of managing incremental buildup on multiple simultaneous fronts simultaneously is magnified by the interactions of the new aspects of the approach.

- Piloting can be an essential learning vehicle in the Plan/Do/Learn cycle. In many cases, for example, the team postulated a process refinement and then piloted the refinement and adjusted it before committing it to practice. There were also some cases where an approach reached the pilot stage only to be abandoned.

2.5 Importance of Process Architecture[1]

Lesson: A sound process reduces a project's need for "heroes".

Lesson: Cooperative relationships among groups are facilitated by well-articulated processes.

Lesson: Review of high level process architecture by key people is critical to the minimization of rework.

We close with these three lessons, since they provide a good transition into the remainder of the paper.

The first two lessons illustrate the project's appreciation of the value of a defined process. The third lesson calls attention to the importance of project agreement on the top-levels of the process - termed the "process architecture".

It is interesting to note that this third lesson was actually recorded after year one of the three year project. At the close of the project there are still significant aspects of the approach that are ill-defined - particularly those related to the product-line-wide activities, such as Domain Engineering. Thus, the organization continues to grapple with front-end process definition issues.

In fact, the authors contend that any organization endeavoring to keep pace with technology will be forced to cope with front-end process definition issues until they close their doors. Our belief is that focusing on Process Architecture is key to managing the evolution of the organization's approach - that is to say, managing the continual transition from vague concepts to defined process.

---

1. At the outset, the Demonstration Project had a very loose notion of Process Architecture. The authors' current definition of the term (based in part on our process experience on this project) is provided in Section 3.5, Create/Update the Process Architecture, on page 14.

3.0 Approach Definition Guidelines

This section offers a structured set of guidelines on how to go about building up your approach. The premise is that your organization:

- knows the value of a defined process,
- appreciates the risks associated with major change/reengineering, and
- has decided that it is a necessity to make substantial changes to your way of conducting business (and may even be working out a brand-new process from scratch).

These guidelines are based both on experience and study: project experience (e.g., the Demonstration Project experience summarized in Section 2.0, Lessons Learned), R&D work with other process engineers (most recently on the STARS program and with the SEI), and study of the process engineering literature.

We must emphasize, however, we do not claim extensive experience in actually applying these guidelines. Rather, we are communicating what we would take into account if we were involved in another project engaged in significant Technology Transition.

3.1 Approach Definition: An Integral Part of the Software Process Lifecycle

Figure 1 depicts the Software Process Lifecycle - showing a single activity called "Approach Definition", which yields the Process Architecture.
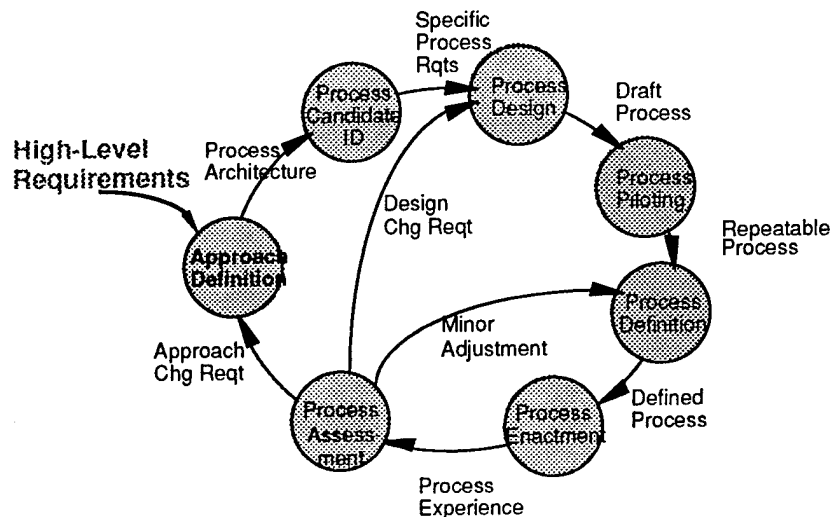

Figure 1. Software Process Lifecycle

3.1.1 "Approach Definition" = "Process Architecture Definition"

We are recommending an *incremental, iterative* buildup of the approach. Thus, we do not envision one pass through Approach Definition, after which time the Process Architecture is complete and stable. Instead, we see successive passes, with the architecture being solidified first in some areas, then in others - and with a refinement of process definition occurring at each iteration.

By our definition, then, the rate of change of the Process Architecture is one good indicator of how well the organization understands the approach...*or at least this would be true for an organization that is serious about process.* (For an organization not serious about process, there will probably be no way to tell whether your approach is solidifying or not!)

Even when the Process Architecture is stable, there may be many details of the approach left undetermined; but we would insist that there must be organizational consensus that those details, once resolved, will have little effect on other aspects of the process.

What do we mean by Process Architecture? This is a difficult question, and there is certainly no industry consensus on the meaning of the term. In our opinion, a useful working definition is the following:

Process Architecture is a high-level description of the process that captures the key aspects of the organization's approach in a form understandable to both stakeholders and participants.

Refer to Section 3.5, Create/Update the Process Architecture, on page 14, for a more concrete definition.

Since this lifecycle is evolutionary (i.e., incremental, and iterative), we see Approach Definition producing incremental updates to the Process Architecture, as shown in Figure 2.
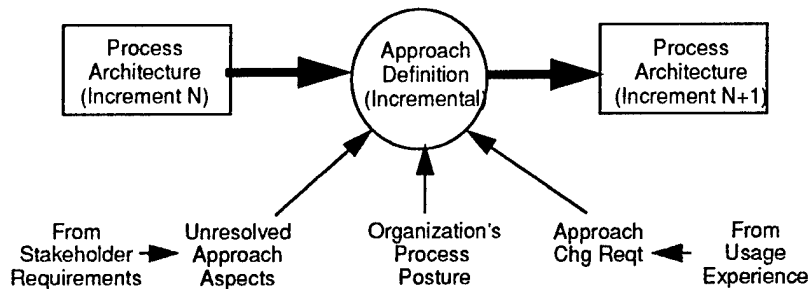

Figure 2. Incremental Approach Definition

The influences on Approach Definition depicted in this figure are:

- the required changes to the already-defined portion of the approach, based on usage experience in the context of the process defined using the prior Process Architecture increment;
- the unresolved aspects of the approach which have not yet been addressed in the organization's process; and
- the organization's posture with respect to handling all of the targeted changes - a multi-dimensional assessment including current expertise, resources available to work on the process, process maturity level in general, etc. (Bear in mind that throughout the transition process, organizational and cultural forces have the potential to reorient, redefine, or totally derail your new approach!)

3.1.2 High-Level "Approach Definition Process"

Given these influences, Figure 3 depicts the composition of this incremental Approach Definition activity. The guidelines provided in Sections 3.2 through 3.6 are organized according to these steps.

Analyze/Update The Process Requirements *Purpose: Maintain a clear statement of the requirements on what the process needs to do, how it should do it, and how it should be represented. These requirements stem from the organization's stakeholders and from usage experience with the process to date.*	See Section 3.2 (page 9)
Assess The Organization's Posture *Purpose: Realistically assess the organization's capacity to further define/improve the approach at this point in time.*	See Section 3.3 (page 10)
Plan First/Next Process Architecture Increment *Purpose: Identify high-priority process engineering tasks for this process architecture increment, based on analysis of the process requirements, the organization's posture, and meaningful assignments of "process element groups" to engineering teams.*	See Section 3.4 (page 12)

Create/Update The Process Architecture
 	
Purpose: Improve the articulation of the organization's approach by improving the process architecture in accordance with this increment's plan.
 	
This is done by carrying out the engineering tasks identified in the process architecture increment plan:
 	
for each process engineering task
    Engineer "Process Element Group"
        Purpose: Add, refine, restructure (or possibly even delete) selected portions of the process architecture, using pragmatic engineering with strong emphasis on piloting.
	See Section 3.5
	(page 14)
	
	See Section 3.6
	(page 18)

Figure 3. Incremental Approach Definition Process

The remaining sections provide guidelines for the above steps. At the start of each section, we the following icon to remind the reader of the context:

analyze/update the process **requirements**
assess the organization's **posture**
plan first/next architecture increment
create/update the process **architecture**
engineer "process element group"
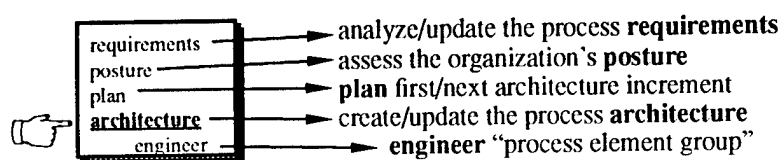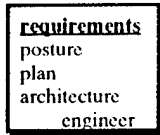
3.2 Analyze/Update the Process Requirements

Purpose: Maintain a clear statement of the requirements on what the process needs to do, how it should do it, and how it should be represented. These requirements stem from the organization's stakeholders and from usage experience with the process to date.

When you already have a process in hand and are making incremental improvements, the requirements are already fairly well stabilized. When you're trying to make major approach changes, it may be a shock to realize all of the sources of requirements you need to take into account. Just as with a software system, you need to take care to identify, organize, and prioritize your process requirements. Here is a checklist to consider:

## 3.2.1 Stakeholders: the Ultimate Source of All Requirements

Requirements come from stakeholders - sources of funding and support, or whose interests are of some concern to you. You need to be clear about their expectations and needs so that you can organize them, prioritize them, and cross-check your resulting process against them. Here is a partial list, based on our experience:

- **Customers:** Presumably your organization is providing a service to or building products for customers. Competitive pressures may be the main impetus for changing your process. In addition to your own internal process goals to address efficiency and productivity, you may need to address specific customer issues - such as whether your organization has been certified as conforming to ISO-9001.

- **Funding Sources:** Funding sources might not just be customers. Examples of other funding sources are company (or DoD) sponsored R&D initiatives - such as the ARPA/STARS initiative that formed a partnership with the Air Force for the Demonstration Project. In fact, the Demonstration Project organization obtained funding from several DoD sources, each of which had expectations that came along with their support - most of which had direct impact on the process.

- **Peer Organizations:** Your organization may be part of a larger whole, and other parts of the organization undoubtedly have explicit (and implicit) requirements on how you do your work. If you fail to provide a required configuration package, for example, an external Quality Assurance department may refuse your delivery.

- **Upper Management:** Review their expectations; accept those you can't change, chip away at those you can, but only if they are really important. Such requirements could run the gamut from addressing specific chronic quality issues, to providing certain metrics, to taking on additional responsibilities.

- **Higher-Level Organization Mandates:** An example of such a mandate might be a DoD directive that all service organizations of a certain type achieve an SEI CMM Level 3 maturity rating within two years. Affected organizations would be required to demonstrate that their process addresses the requirements for the corresponding Key Process Areas [SEI-CMM93, SEI-SPF-94].

- **Your Own Organization's Goals:** Your organization is also a stakeholder. Among other things, you need to take all the other stakeholder requirements into account and formulate your own statements of vision, mission and goals that will guide the organization's growth. The Air Force/STARS Demonstration Project organization accepted a mandate from the USAF HQ and from the STARS Program to lay the groundwork for a new product-line process. As summarized in Table 1 on page 2, this mandate translated into a lower-level set of technology goals representing a strategy for getting there.

- **Your People:** Your people will have to run the process. If they are disenfranchised, the process will fail. Conversely, if they are enthusiastic owners, the process will thrive. Your people provide important requirements not only on the content of the process, but its form of representation and level of detail.

3.2.2 Types of Requirements

With respect to front-end process definition, requirements fall into two broad categories: process content (what the process should do and how well it should do it), and process representation (notation, structure, level of detail, etc.). Here are some typical requirements from our experience:

- **Examples of Requirements on Process *Content***
  - Increase productivity and efficiency; decrease cycle time.
  - Use a particular set of new technologies (e.g., Cleanroom, OO).
  - Build applications using a common software architecture; and set up a group with technical responsibility for articulating, promulgating, supporting and evolving the architecture.
  - Increase the level of automation (possibly targeting specific areas, such as requirements traceability).
  - Build up and nurture clusters of expertise in certain areas (domain, technology, or other).
  - Provide more metrics to help track progress and quality.
  - Adhere to a particular lifecycle model (e.g, spiral).
  - Achieve a high degree of reusability; minimize redundancy; provide tailorability of key process elements.
  - Focus on walkthroughs as a way to improve product quality.
  - Respond to specific standards and policies (e.g., TAFIM).
  - Give first priority to the products delivered to customer X.
  - Use good encapsulation techniques (e.g., implement a layered software architecture).

- **Examples of Requirements on Process *Representation***
  - Provide views which address the needs of specific audiences (e.g., newcomer, practitioner, process engineer, manager).
  - Use a specific, mandated notation (e.g., $IDEF_0$).
  - Use a specific process support tool (e.g., DesignIDEF by MetaSoft or PEAKS by ccPD).
  - Use a template approach to provide consistent, prescribed information for key diagram elements (e.g., PDIOTs [PhillipsEtt95]).
  - Define processes to the "enactable" level in certain areas (e.g., software increment development).
  - Clearly identify where metrics (process and product) are collected and used.
  - Designate risky areas of the process and discuss what is being designed in to mitigate them.
  - Give first priority to the process concerns of customer Y.
  - Adhere to the quality criteria listed on page 16.
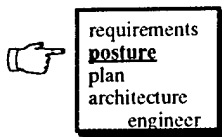
3.3 Assess the Organization's Posture

☞ requirements **posture** plan architecture engineer

Purpose: Realistically assess the organization's capacity to further define/improve the approach at this point in time.

This step is particularly important at the start of a major initiative, although it should be revisited at the start of each incremental step. How you start the ball rolling can have a great impact on its subsequent direction and momentum. Carefully analyze your status and capability from a number of standpoints, and form a realistic assessment of the number of simultaneous changes you are attempting and their significance, and balance that assessment against your capability to handle those changes. Reviewing the guidelines in this section may help weed out unreasonable objectives early.

Here is a partial checklist of factors you may wish to consider:

- **Maturity of your existing process**: if you already have a sound, defined process, you will be in a position to better assess the specific areas of impact. Begin by identifying areas of your process you will be able to recover with little, or no change, and then focus on the remainder.

- **Existing process engineering experience**: an organization with a CMM Level 3 maturity rating, for example, will have a leg up on an organization that doesn't really understand what it takes to have a defined process. If your organization is rated at Level 1 or a low Level 2, we strongly recommend you first do a stand-alone process engineering pilot using a number of people with leadership skills. When a viable process definition approach is established that makes sense for your organization, use these people as "change champions" to define and transition the approach into the organization.

- **On-board expertise in the technologies**: note that while training is useful, piloting is much better, and real project experience is "golden". If expertise is a key problem, consider aggressive staffing measures - possibly including consulting help.

- **The extent of the change(s)**, including the *number of new technologies; degree of divergence* from existing methods; *how pervasive the changes are* in terms of the number of process aspects you will have to impact; *how interrelated the technologies are* - and if they are interrelated, *how much potential dissonance there is* among the technologies. These are admittedly very subjective assessments - especially if your level of experience in the technologies is limited! Regardless of your assessment, be aware of the risk you are taking on. The transition to any new technology is risky (resulting in failures in a majority of cases), and transition to multiple technologies compounds that risk.

- **Degree of challenge of the real project(s) you will be undertaking as vehicles for "proving-in" the new process.** Piloting only takes you so far in establishing your new approach; in the end you simply have to give it the acid test. While you are attempting to perform on the new project, you'll be finding out all the ways the approach doesn't really work very well and has to be adjusted or scrapped. But if your key people are wrapped around the axle learning a new application domain for the system they're building, or if they are slaved to an overly ambitious schedule, how will they have time to adjust the approach? If your sights are too high, you will either fail in building the system (and thus fail in the TT), or your team will abandon some or all of the new approach and fall back to their old heroic ways (and you've still failed in your TT - perhaps without even knowing it!).

- **Willingness/eagerness of personnel (including management) to accommodate/integrate change.** This is a multi-dimensional issue - on the individual, organizational, and cultural levels. There are many TT references that may be of use in forming this assessment, such as [SPC-TTG94, SEI-MTC96].

- **Number of participating organizations and their track record for working well together.** If there is no track record, it's safe to say there **will be** problems. On a related note, if some of the organizations already work well together, there is a risk that newcomers will be shunned as outsiders.

- **Extent of other, potentially competing aspects of TT, such as metrics** - which can take an amazing amount of resources - and which will be severely affected by the new aspects of the process.

- **Number of new requirements on the process engineering itself** being imposed by stakeholders - such as level of detail, required tools to be used, intent to prove automated enactment Treat process definition like systems development: clearly identify and justify all the requirements on the process and maintain requirements control.

- **Number of stakeholders[1] with potentially divergent goals.** The stakeholders will be making demands along the way. Negotiating these demands, and possibly resolving conflicts with the stakeholders - who might just have some *conflicting demands* - can take quite a bit of your **intellectual bandwidth**.

In the authors' experience, "intellectual bandwidth" (IB) is a key factor in your likelihood of success in making your targeted transition. We define it (very loosely) as follows:

> *IB is the aggregate capability of your most creative and productive people minus all of the demands on their time for everything but what you really need them to do.*

You really need IB when you're defining your new approach. The above bullet cites just one of the drains on IB. The next bullet is another - one which is impossible to appreciate unless you've lived through it at least once.

- **Degree to which you will have to service external TT requirements:** on the Demonstration Project - which was set up explicitly to accelerate TT across a wide community - this profound demand on IB has led one of the authors to coin:

  Randall's Law: *The more progress your organization makes in figuring out a new and better way of doing business, the more time your key people will spend explaining your approach - and the less time they will have to contribute to figuring out the new way of doing business.*

---

1. By "stakeholders" we mean the folks paying the bills or to whom you have some obligation. Many organizations have a number of stakeholders - particularly if your funding has been scratched together from an array of sponsors/participants.

Corollary 1: *There is a natural cap on the rate of forward progress in figuring out a new and better way of doing business.*

requirements posture **plan** architecture engineer
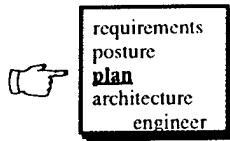
3.4 Plan First/Next Process Architecture Increment

Purpose: Identify high-priority process engineering tasks for this process architecture increment, based on analysis of the process requirements, the organization's posture, and meaningful assignments of "process element groups" to engineering teams.

Planning the first/next increment of your Process Architecture is tricky business, because it involves so many unknowns. We can offer some suggestions about how to select among the various not-yet-defined aspects of your approach, but one of the key complications is that the prioritization depends to quite an extent on the Process Architecture you already have defined in prior increments.

In a way, this situation is similar to that when a software project has to respond to a major change to its requirements - such as an ECP to an existing system. Your initial analysis is simply to understand the requirement, but to really see what the impact is you have to look at the software architecture.

One thing the architecture does is help identify which groups need to get involved to see what those impacts are. This is because - as we will see in the next section - a good architecture will include a mapping to the organizational group(s) responsible for the corresponding part(s) of the architecture. Thus, once you have an idea which parts of the architecture might be impacted, you can refer to the mapping to see who to call into the room.

In this section, we provide some high-level pointers on factors take into account when judging which aspect(s) of the approach you should tackle next. We also discuss how to identify manageable process engineering tasks to assign to teams.

3.4.1 Pointers on Judging Approach Definition Priorities

- **Urgency of Need for Project Work**

  By this point in the paper, it should be clear that we think you'll need *real* project experience to build up a *real* process. Particularly on the first real project, you'll also be "staging" your approach definition as you go. Thus, you'll need to identify which aspects of your approach are "coming due" the soonest. Examples:

  - If you're about to start a specification effort and you think you might want to use Cleanroom Software Engineering [Cleanroom96], this might be a great time to think about how Cleanroom might effect your process[1].

  - If you're going to deliver your first configuration of software to the customer's CM organization in three months, but you haven't figured out how to baseline OO classes or how to explain to the receiving organization what they even mean, you might want to raise the priority on this aspect of the approach.

- **Urgency of Need for Purchase or Contract Decision**

  Acquisition cycles can take annoyingly long - sometimes amazingly long. You need lead time. So your approach needs to be ready before you will be applying it. Examples:

  - Which OO modeling tool did you want, and how much processor horsepower will it be taking away from your Ada development environment?

  - You need to issue a Statement of Work for a contract you want to start in four months - so are you going to use OO or not? If so, what contractual wording do you want to put in about QA, CM, standards, etc. Or do you "wing it"?

- **Risk to Process Architecture**

---

1. Hint: there will probably be more than just your specification process involved before you're done realizing the benefits of Cleanroom!

Now we're thinking like software engineers! Pay attention early to those requirements that could have the biggest impact on your Process Architecture. The Architecture has a lot to do with the way all of your process elements work together, and if you think there might be some structural changes to accommodate a particular technology, this is one reason to give it early attention.

- **Level of Stakeholder Concern**

  Going back to the ultimate authority of what's important to your organization, you had better give the stakeholders some say in your prioritization.

- **CMM/ISO 9001 Exposure, etc.**

  Your attention to the SEI CMM or ISO 9001 evaluation criteria may be due to several reasons - from responding to stakeholder demands to attempting to study your process using well-thought out references as tools. Whatever your reason, these references may cause you to overhaul your approach. For example, looking at the Software Configuration Management Key Process Area of the CMM may motivate a complete overhaul of your CM. This might, in turn, affect so many parts of your process that you decide to take a good look at your architecture - to pull out all of the CM aspects from the places in which they are currently ferreted away, and to define a common set of CM subprocesses that serve all process elements.

  Something with such widespread impact may call for a high-priority. In fact, despite conventional wisdom that "CM is trivial", you may actually assign a lead person or two to work on the engineering!

3.4.2 Mapping Aspects of the Approach to "Process Element Groups"

Having assessed the Approach Definition priorities, as discussed above, management must select which approach aspects to tackle for the next increment. This involves determining what expertise is available and their competence to address the engineering needs, but it also involves identifying meaningful groups of "process elements" - based on the current Process Architecture - for actual teams to work on.

We use the term "process element" to refer to a process model abstraction consisting of a set of related activities, artifacts and agents (hopefully adhering to the encapsulation quality criteria discussed in Section 3.5.3, Process Architecture Quality Factors, on page 16.) When building up a new approach, process engineering tasks often deal with several process elements at once. This is because a targeted new method or technology may affect several aspects of the Process Architecture.

For example, if an organization seeks to transition from a functional-decomposition software paradigm to an object-oriented software paradigm, there will likely be significant impacts to many existing process elements - such as design, implementation, configuration management, and project management. It is also possible that some existing process elements will be discarded entirely and that the process architecture will be significantly restructured.

In keeping with the loose "approach definition process" outlined at the beginning of this section (refer back to page 6), we recognize that a process engineering task may involve several process elements - and that there may be several process engineering tasks active at one time.

Just as with software development, we try to choose our tasks to balance numerous factors, such as priorities, personnel availability, and "orthogonality" (i.e., we try to have the tasks working on segregated aspects of the architecture, although the tasks may need to closely coordinate with respect to interfaces). Figure 4 depicts one step in a hypothetical iteration of a process architecture. There are two aspects of the approach that are deemed high priority for this iteration. Approach Aspect P seems to impact process elements in the top layer of the architecture, whereas Approach Aspect Q impacts process elements in the top two layers of the architecture. The organization has decided to allocate the refinement work to two teams, but has opted to align their tasks along architectural lines. The two teams must coordinate their work with respect to Approach Aspect Q. The result of the refinement work requires an adjustment to the structure of the top layer of the architecture and the addition of a new process element to the second layer of the architecture.
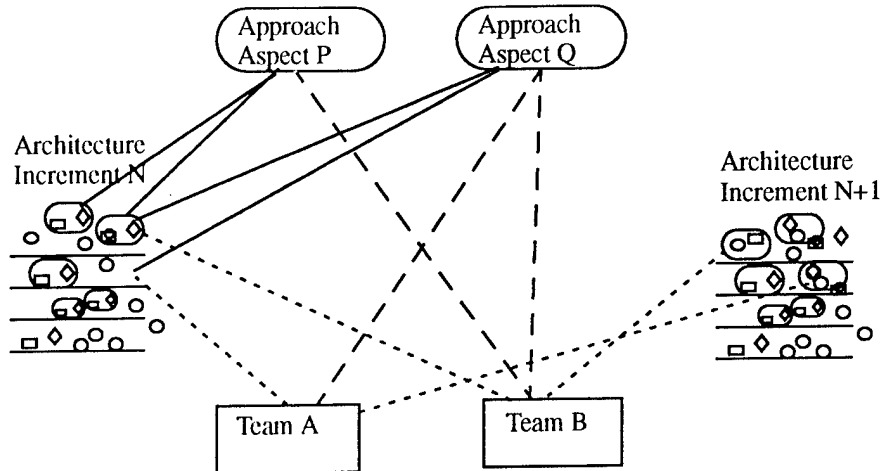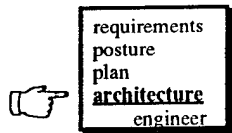
Figure 4. An Approach Definition Increment

3.5 Create/Update the Process Architecture

| requirements |
| posture |
| plan |
| **architecture** |
| engineer |

☞

Purpose: Improve the articulation of the organization's approach by improving the process architecture in accordance with this increment's plan.

In Section 3.1.1, "Approach Definition" = "Process Architecture Definition", on page 6, we emphasized the importance of Process Architecture, which we defined as follows:

> *Process Architecture is a high-level description of the process that captures the key aspects of the organization's approach in a form understandable to both stakeholders and participants.*

In this section, we provide a more concrete definition of Process Architecture, drawing on the notions of software architecture as a starting point. We then attempt to flesh out the definition with some quality criteria - which we hope will provide guidance to organizations who are formulating their own Process Architecture.

According to Osterweil, "Software Process is Software, Too" [Osterweil87]. The authors agree with the spirit of this assertion: our experience leads us to believe that the analogy with software has a great deal of strength. Even when we do not have a very good idea of how to systematize our approach, software systems engineering techniques are useful.

3.5.1 The Notions of *Software Architecture*

Pursuing the idea that software engineering techniques apply to systems of processes, it is useful to examine what software experts are saying about Software Architecture - since this may give us insight into Process Architecture. Interested readers may wish to browse the Software Architecture overview in SEI's Technology section on the WWW [SEI-Arch96]. This overview includes recent quotes from a number of well-known authors, such as the following:

Boehm, et al., 1995:

> A software system architecture comprises
> - A collection of software and system components, connections, and constraints.

- A collection of system stakeholders' need statements.
- A rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements.

These quotes are sufficient to show both the common elements of the notions and some of the embellishments. What we cull from the literature is the following definition - which we have augmented based on our own experience with a large spectrum of software systems:

Software Architecture

A model of a software system that provides:

(1) A description of the high-level structure of the system, including

 > The composition of the system's data and processing components, and
 > The relationships and interconnections among the components.

(2) Guidance or rules on how new components should be added to the architecture, and how the architecture should be evolved over time.

(3) Rationale for why systems built using the architecture would satisfy stakeholder requirements.

(4) References to any standards or methods that are tied to or assumed by the architecture, whether mandated by stakeholders or chosen as foundation principles.

(5) A description of the high-level dynamic behavior of the system, showing how the components would work together and synchronize their work over time to satisfy end-user scenarios. This would be especially important for large complex systems where behavioral issues (high-level state transitions, performance characteristics, etc.) were critical for understanding and guiding the system's construction.

(6) A refinement of the structure and behavior showing allocations and relationships to physical hardware (processors, networking, etc.). This would be especially important if a common platform strategy is desired.

3.5.2 The Notions of *Process Architecture*

The literature on process technology is relatively new, and there is even less agreement on the definition of process architecture than there is on software architecture.

Here are excepts from Feiler's and Humphrey's well-known paper dealing with software process terminology [FeiHum92]:

Process Architecture

A conceptual framework for consistently incorporating, relating, and tailoring process elements into enactable processes. An architecture provides a space of process designs. A process architecture is often needed when a process must relate to other existing or future processes. Examples of such needs are process element reuse, process enactment, and process tailoring. An essential property of a process architecture is its ability to indicate whether a process element is or is not compatible with the architecture.

Process Element

A component of a process. Process elements range from individual process steps to very large parts of processes. They may be templates to be filled in, fragments to be completed, or abstractions to be refined.

While we agree with these notions in principle, we believe there should be a more concrete definition of process architecture, similar to those for software architecture, above.

Here, then, is our working definition of process architecture, which takes the same form as the definition of software architecture provided on page 15:

Process Architecture:

A high-level model of a process which includes descriptions of:

(1) Static structure of its composition, in terms of its Artifacts, Activities and Agents[1] and their interrelationships.

(2) Guidance or rules on how new process elements should be added to the architecture, and how the architecture should be evolved over time.

(3) Rationale for how the process architecture addresses the needs of the stakeholders.

(4) References to any standards or methods that are tied to or assumed by the architecture, whether mandated by stakeholders or chosen as foundation principles.

(5) High-level dynamic behavior of the process, showing how the components work together and synchronize their work over time. This is especially important for large, complex processes, but it may be deferred to later stages of process design, at the discretion of the architect.

(6) A refinement of the structure and behavior showing allocations and relationships to actual organizational elements. This is especially important if it is desired to define a common organizational structure for all executions of the process. (Note that our use of the term "agent" as used above does not necessarily require specifying the agent's organization.)

3.5.3 Process Architecture Quality Factors

Although we will be offering a few examples to illustrate the notion of process architecture, we do not wish to specify a notation. Notations are best chosen by the organization, based on the nature of its process, the preferences of its process engineers and its practitioners, mandated toolset, etc. Instead, we offer a set of quality guidelines to serve as a checklist of considerations to take into account regardless of your notation. Many of these guidelines apply equally well at lower levels of the process.

- **Completeness**

    The process architecture should cover the aspects listed above.

    Each aspect described by the model should be well-defined. Some authors suggest using templates to capture pre-defined information, but volume and level of detail is a critical issue, particularly at the architecture level.

    Views should be provided to address the needs of key audiences (especially the needs of the stakeholders), including these categories: newcomer/outsider, practitioner, manager, process engineer.

- **Consistency**

    The different views of the model should be consistent with each other. Note that even at the architecture levels of a process model, if we are dealing with a large, complex process there may be a strong need for automated support to construct and maintain the model (i.e., an integrated toolset/repository).

- **Simplicity**

    This principle applies in several senses:
    - **Avoid unnecessary detail**, concentrating on the essential aspects. (This principle applies at the lower levels of the process too: bear in mind that every bit of detail in the model is detail that will be maintained - or discarded.)
    - **Simple views** should be available, to aid understanding (even if the model is complex)
    - **Design elegance**: the process should be well-engineered to select good abstractions that maximize reuse of common ideas and minimize interconnections.

- **Encapsulation and Abstraction Principles**

    We are strongly influenced by software engineering principles here:
    - **High cohesion** within process elements: seek clusters of activities/artifacts/agents that have strong commonality of function and data.

---

1. **Artifacts** are the external products received from or delivered to outside organizations, or internal products developed and maintained as part of the organization's process. **Activities** comprise the work being done to consume/produce the artifacts. **Agents** are responsible for performing the activities (and may be people or programs). This "AAA" terminology is also used by the SEI [ArmKelPhil93].

- **Low coupling** between process elements: seek clusters of activities/artifacts/agents such that separate clusters have few interfaces.
- **Information hiding**: a process element that has no explicit need for an artifact should have no access to it.
- **Layering**: attempt to partition the process elements into abstraction layers, both for ease in conceptualizing and for ease in maintenance.
- **Reuse of common elements**: including identification of kernel or service elements that can be reused throughout the process in different contexts. In identifying commonality, seek opportunities to "tailor" process elements; i.e., instantiate a process element in different ways for different parts of the process. (Note that some authors have recently been urging an object-oriented approach to process modeling [e.g., JacEricJac95] - if successful, this approach would leverage inheritance and polymorphism to facilitate reuse.)

• **Stability**

The degree of change you are experiencing in your architecture will give you insight into two key issues: how rapidly your approach is solidifying, and how well your abstractions have been chosen.

• **Process/Organization Alignment**

We believe a process is best applied if the organization is aligned to it. A well-architected process can be instrumental in molding an organization. (Note that an organization engaged in process reengineering should avoid using existing organizational structure as a major process driver.)

Here are some related thoughts:

- **Integrating with the Parent Organization**

  If the process will be executed by an organization that serves a larger organization, the larger organization is clearly a stakeholder. The "parent" organization imposes a set of external interfaces to the process. It may also impose set of organizational constraints which may affect process architecture. In such cases, it may be possible for encapsulation techniques to preserve architectural integrity.

  For example, Configuration Management (CM) may be required to be administered by a department that reports to a manager outside of this organization. If the new process includes a new object-oriented approach, for example, there may well be fundamental changes to the CM process versus what the parent organization is used to. It may be possible, however, to encapsulate the new aspects of the process while preserving the spirit of the parent process. If so, it would help insulate the architecture from changes from both sides - despite the organizational constraints.

- **Considering Organizational Realities**

  Similar considerations apply to other organizational constraints - such as the apparent necessity to use a particular external group (e.g., contractor or vendor) because of their unique expertise. You may not be able to change the other group's process, yet you want to maximize architectural integrity. Encapsulation techniques seem to apply here as well. Furthermore, our experience tells us that things change. A group whose skills are vital today, might be displaced tomorrow (or go out of business, or whatever). In this case, as in the above case, we would tend to emphasize process architectural integrity first, then try to take organizational "realities" into consideration.

- **Considering Cultural Realities**

  This is admittedly a very difficult characteristic to judge, but when you are trying to make significant changes to your approach, culture dissonance might end up being the most serious risk of all. Technology Transition resources may help - not to only judge the process/culture alignment, but to facilitate organizational and personal adjustment [e.g., SPC-TTG94, SEI-MTC96, Moore95, MangKlein94]. Several factors seem particularly significant in optimizing process/culture alignment:

    Clear process requirements - especially well-articulated goals: process engineering must assure that the requirements are systematized, but management bears the responsibility for the goals - both for communicating them and for living them.

    Buy-in through participation: the process owners are the process users; and

    Incremental buildup: both in process content and in process representation, it is essential to take it a step at a time.
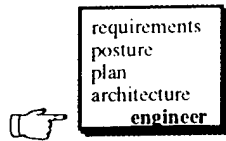
3.6 Engineer "Process Element Group"

```
requirements
posture
plan
architecture
engineer
```

Purpose: Add, refine, restructure (or possibly even delete) selected portions of the process architecture, using pragmatic engineering with strong emphasis on piloting.

Earlier subsections have discussed how the organization identifies aspects of the new approach that make sense to target for this iteration of the Process Architecture. This section now discusses how the engineering would proceed to determine whether the changes should be made - and if so, what those changes should be.

Before launching into a discussion of these engineering steps, however, it is important to explain why "Process Element Groups" come into play.

We are discussing how a team working on such a task might proceed. There are two key themes:

- **The degree of refinement is based on how well the organization understands the approach.** To help make this notion more concrete, we define a set of four "Approach Understanding States" and use these states at key decision points in the refinement procedure.

- **The primary way to increase approach understanding is through experience.** Thus, we recommend that the build up of the approach be incremental, with piloting and real project experience built into each increment.

3.6.1 Approach Understanding States

A key factor to help in deciding what to do next is "how well do you understand your approach". Here is an attempt to define a four-level answer to that question:

- **State 1: Ill-defined Approach** - loose (or non-existent) requirements; interfaces with other process elements sketchily specified; several competing technologies to either adapt/integrate or discard; little (or no) team experience in technologies; disagreement among key project members.

  Action: Regard as high-risk; don't even think about committing to it without future refinement and piloting; if you have many of these at high-levels in the architecture, seriously consider cutting back on your ambitions (i.e., abandoning some of the planned aspects of the approach); potentially the biggest risk: the CM aspects of your process, which cannot be finalized until most process elements are beyond this state.

- **State 2: Basic Understanding of Approach** - stable requirements; technologies are adapted/integrated; some team experience in the technologies (via piloting - or preferably, via actual project experience); interfaces with other process elements well-specified; agreement among key project members.

  Action: Regard as moderate risk; keep an eye on it while you ease into operational usage; form an understanding of the key issues and put metrics in place to track; begin working out process-driven planning.

- **State 3: Operational Competence** - approach well-understood and being successfully practiced for production work; interfacing process elements work smoothly with this area; management views activities as low-risk and has high confidence in status assessments; suitable for process-driven planning; ready for tailoring for other projects.

  Action: Regard as low-risk; emphasize process-driven planning; selectively refine portions of the process definition to the enactable level; emphasize improvement cycle.

- **State 4: Detailed Understanding** - most activities are rote; some (or all) susceptible to automated enactment; metrics-based product and process improvement routine. (Note: some process elements will never achieve this state, even for CMM level 5 organizations, since some activities inherently depend on creative responses.)

  Action: Refine process definition to enactment level as appropriate; consider automated enactment support (but don't rest on your laurels: this may be the next area to be hit by paradigm shift).

This paper is focusing on getting from State 1 (ill-defined approach) to State 2 (basic understanding). Clearly, we recommend deferring formality and detailed exposition until the organization has achieved at least State 3 (operational competence).

3.6.2 The Process Element Group Engineering Procedure

Given the above notions of Approach Understanding States, we are ready to provide additional insight into the engineering activity.

> **determine** the Approach Understanding State and the current level of process definition
> **if** Approach Understanding State is already sufficient for refinement **then**
> > **refine** the process elements one or more levels, as appropriate
> > **recommend** updating process architecture accordingly without further evaluation
> **else** *[Approach Understanding State does not yet support refinement]*
> > **assess** reasonableness of gaining sufficient understanding for this iteration
> > **if** unreasonable **then**
> > > **if** decision about approach aspect can be deferred, **then**
> > > > **defer** until next iteration *[flag as major project risk]*
> > > **else abandon** approach aspect
> > **else** *[reasonable to gain more understanding; so try to get some now]*
> > > **take a stab** at one or more refinement steps, as "appropriate"
> > > > *[see "Brainstorming" procedure, detailed below]*
> > > **review** with other process elements and improve
> > > **construct strategy** and plan for piloting/adopting
> > > **conduct pilot**(s) for key areas
> > > **review results** of pilot and improve
> > > **if** this rendition seems to work, **then**
> > > > **assess risks** and insert metrics/controls as needed for early prove-in phase
> > > > **define training** needed for prove-in phase
> > > > **recommend appropriate updates** to the baselined process architecture
> > > **else**
> > > > **if** decision about approach aspect can be deferred **then**
> > > > > **defer** for another stab on a later iteration *[flag as major project risk]*
> > > > **else abandon** approach aspect
> > > > > *[Note: this may have significant ripple effects on other process elements, or even process architecture... but good architecture should help.]*

At this point, your team has completed the engineering for this Process Architecture increment and has done the following for each of the targeted changes:

- **Thrown it out** as incompatible or too risky for now *[whew!]*
- **Deferred it** 'til the next increment *[wait a minute...I thought this was really high priority!?]*, or
- **Established sufficient confidence to make to the architecture change** - and set the organization on a course toward the associated process definition *[...hope these guys know what*

Figure 5. The Process Element Group Engineering Procedure

3.6.3 The Brainstorming "Subroutine" for a Defining/Refining a Process Element Group

This is embedded in the above "Process Element Group Engineering" procedure:

- Review the stakeholders. This is really the first step in making sure you understand the requirements for the process element - and it depends on the level of the model.

4.0 Summary

The authors believe that there are some distinct process development problems faced by organizations attempting to make *revolutionary* changes to their existing way of doing business (reengineering) - or who are attempting to start a new operation from scratch. In our experience and in our review of the process-oriented literature we have found little guidance for such organizations. These problems are often quite different in character from those faced by organization's engaged in *evolutionary* process improvements - or who are attempting to capture their understanding of their existing process.

Section 1.0, Introduction, provided the context of the Air Force/STARS Demonstration Project, from which much of the authors' experience has stemmed. This project, charged with spearheading a new product-line process, was (and is still) engaged in significant technology transition. Section 2.0, Lessons Learned, cited some of the applicable lessons learned from this project - which are especially instructive because of the level of attention that has been given to process.

Our experience has led us to formulate the Approach Definition Guidelines outlined in Section 3.0 The guidelines are keyed to the steps in a high-level *approach definition process*, as follows:

- Analyze/Update the Process Requirements (Section 3.2 on page 9),
- Assess the Organization's Posture (Section 3.3 on page 10),
- Plan First/Next Process Architecture Increment (Section 3.4 on page 12),
- Create/Update the Process Architecture (Section 3.5 on page 14), and
- Engineer "Process Element Group" (Section 3.6 on page 18).

To recap our major recommendations:

- Emphasize Process Architecture as instrumental to managing the convergence on your new approach (we provide a definition of Process Architecture, and we offer a set of quality criteria).
- Use an incremental, iterative method to build up the approach - i.e., to build up your Process Architecture. Plan each increment based on:
  - assessments of the priorities of unresolved aspects of your approach (based in turn on stakeholder requirements and on your own experience from using the process to date), and
  - assessments of how much your team is able to really accomplish at this point in time.

  These assessments may lead to deferring (or totally abandoning) some targeted aspects of your approach, but the aspects that survive will have a better chance of surviving into actual usage - and into full-fledged process definition.
- Value actual hands-on usage experience above all other assessments of viability, and use piloting to get early experience before committing the organization to a new aspect of the approach. Piloting is admittedly expensive (in terms of time and resources), but it can be dramatically less expensive than premature adoption.

It is our hope that our lessons learned and suggested guidelines will not only prove useful to organizations with similar challenges, but will contribute to the vital discipline of process engineering.

References

[ArmKelPhil93]   Armitage, J., Kellner, M., Phillips, R., **Software Process Definition Guide**, Software Engineering Institute Technical Report CMU/SEI-93-SR-18, August 1993.

[CFRP95]   Creps, R., Davis, M., Simos, M., Collins, P., Wickman, Maj G., **Using a Conceptual Framework for Reuse Processes as a Basis for Reuse Adoption and Planning**, Proceedings Seventh Annual Software Technology Conference, Salt Lake City, UT, April 1995.

[Cleanroom96]   **Cleanroom Software Engineering** - a Tutorial on the WWW, as of 1/17/96:
http://source.asset.com/stars/loral/cleanroom/tutorial/cleanroom.html *
This tutorial was prepared for the STARS Program by Software Engineering Technology, Inc., under contract to Loral Federal Systems.

[CroakTackett96]   Croak, LtCol T., Tackett, B., **A State-Based Process Model for Software Development**, Proceedings Eighth Annual Software Technology Conference, Salt Lake City, UT, April 1996.

[DemoExp96]   **AF/STARS Demonstration Project Experience Report, Version 3.0 (Draft)**, CDRL Sequence A011-002D, Electronic Systems Center, AFMC, USAF, December 1995 (available in second quarter 1996; may be accessed via
http://source.asset.com/stars/loral/home.html *

[FeiHum92]   Feiler P., Humphrey W., **Software Process Development and Enactment: Concepts and Definition**, Software Engineering Institute Technical Report CMU/SEI-92-TR-04, September 1992.

[MangKlein94]   Manganelli, Klein, **The Reengineering Handbook**, AMACOM, 135 West 50th St., New York, NY 10020, ISBM 0-8144-0236-4, November, 1994.

[JacEricJac95]   Jacobson, I., Ericsson, M., Jacobson, A., **The Object Advantage. Business Process Reengineering with Object Technology**, Addison Wesley, 1995, 348pp. ISBN 0-201-42289-1.

[Moore95]   Moore, G., **Crossing the Chasm**, Harper, October, 1995.

[Osterweil87]   Osterweil, L., **Software Process is Software, Too**, Proceedings of the 9th International Conference on Software Engineering, Monterey, California, 1987.

[PEAKS95]   **Process Engineering Analysis and Kernel System**, product brochure available on the WWW (as of 1/17/96) at:
http://www.stars.ballston.unisysgsg.com/Newsletters/1995-03/7.html *

[PhillipsEtt95]   Phillips, R.W., Ett, W. H., **Instructions for Defining Processes Using Information Organizer Templates, Version 4.0**, published jointly by the Software Engineering Institute (SEI), and Loral Federal Systems, available from the SEI as SEIM Solution Prototype, IOT-1, May 22, 1995. (This report has a status of "Comment Draft".)

[Randall95]   Randall R.L., Ett W.H., **Using Process to Integrate Software Engineering Environments**, Proceedings Seventh Annual Software Technology Conference, Salt Lake City, UT, April 1995.

[SEI-Arch96]   **"What is Software Architecture"**, from the Software Engineering Institute's Technology section on the WWW; on 1/13/96, this was
http://www.sei.cmu.edu/technology/sw.arch.definitions.html *

[SEI-CMM93]   Paulk, M.C., Weber, C.V., Garcia, S.M., Chrissis, M.B., Bush, M., **Key Practices of the Capability Maturity Model, Version 1.1**, Software Engineering Institute Technical Report CMU/SEI-93-TR-25, 1993. Available on the WWW as of 1/17/96 as:
http://ricis.cl.uh.edu/CMM/TR25/tr25.html *

[SEI-MTC96]   Based on notes from the SEI workshop, **Managing Technological Change**, Software Engineering Institute.
As of 1/22/96, the SEI was maintaining a workshop description on the WWW at:
http://www.sei.cmu.edu/products/prod.descriptions/mtc.html *

[SEI-SPF94\   Olson, T.G., Reizer, N.R., Over, J.W., **A Software Process Framework for the SEI Capa-bility Maturity Model**, Software Engineering Institute Handbook CMU/SEI-94-HB-1, August, 1994.

[SPC-TTG94]   **Using New Technologies: A Technology Transfer Guidebook**, SPC-93097-N, Version 2, Software Productivity Consortium, 1994.
                As of 1/22/96, the SPC was maintaining a product description on the WWW at:
                      http://software.software.org/dec94/products/tipd.html *

[SPC-PDMG94]   **Process Definition and Modeling Guidebook, Volume I, Concepts and Principles of MPDM**, SPC-92041-CMC, Version 2, Software Productivity Consortium, March 1994.

*   **Note:** WWW locations are unfortunately ephemeral: there is no guarantee that a page that exists on the date of the publication of this paper will contain the same information - or even exist - in the future. Because of the value of on-line access, however, the authors chose to include the reference anyhow - with this disclaimer about its currency.

Appendix A - List of Selected Process Representation Notations

In various places in this paper, we have made reference to particular process representation notations. We have used $IDEF_0$ heavily on the Demonstration Project (and have found it to be useful but far from complete) as well as ETVX (similar comments apply).

In this appendix, we provide a brief list of some of the more common notations, with our comments about their nature and applicability. Some of the notations are part of a family of notations connected with a particular method (such as SPC's MPDM).

We are not recommending one notation over another, but we thought readers might find the list (and the references to some associated information sources) useful.

- Integration Definition for Function Modeling ($IDEF_0$)

  $IDEF_0$ [FIPS 183] is based on SADT™ (Structured Analysis and Design Technique™). An $IDEF_0$ model consists of a hierarchical series of diagrams, text, and glossary cross-referenced to each other. The two primary modeling components are functions (represented on a diagram by boxes) and the data and objects that inter-relate those functions (represented by arrows). The $IDEF_0$ methodology also prescribes procedures and techniques for developing and interpreting models, including ones for data gathering, diagram construction, review cycles and documentation. The simplicity of $IDEF_0$ comes in many ways from the constraints of a single view of the system being described. This can be a doubled-edged sword as users try to capture things that do not fit into the model easily.

- Entry/Task/Validation/eXit (ETVX)

  An ETVX [RadPhil 88] model is expressed as a set of interconnected activities for each there must be four aspects; entry (E) and exit (X) criteria, work to be accomplished (T), and the work needs to be validated (V). The model indicates the relationships and flow among the four aspects of an activity and between activities. Activities can be nested thus ETVX can be applied to as low a level as required to control the process.

- Process Definition Information Organizer Templates (PDIOTs)

  The PDIOTs [EttPhillips 95] provide a vehicle for collecting and organizing information necessary to create a process definition that is "enactable" with a desired set of behavioral characteristics. The focus is on Activities, Artifacts, and Agents with efforts to capture ETVX information. When completed with instance names of artifacts and agents, the templates constitute a primitive, but enactable, instance of a process enactment guidebook ready for "manual" enactment.

- Petri nets

  A Petri net [Resieg82] is a directed graph interconnecting nodes of different types on which tokens are permitted to travel. While Petri nets consist of a small number of elements and the algorithms for evaluation can be expressed very simply, they are sufficiently formal to allow mathematical analyses. These tend to be both detailed and complex.

- State Transition Diagrams (STDs)

  STDs, and a more sophisticated variant, Harel Statecharts [Harel87] are usually focused on a single component of the process. The STD depicts the states the component may exist in and the events that cause the component to transition to another valid state. The interactions between components is not visible unless they are mentioned in the events. In addition to the graphic notation it is also straightforward to represent STDs as tables.

- Flowcharts

  Possibly the original process notation, it captures activities their sequencing and the decision points affecting the process flow. Its use is normally restricted to detail levels. It is easy to use and simple to understand but remains very informal.

- Rummler Brache Notations

  The focus of Rummler Brache [RumBrach90] is process capture and improvement. The diagrams are deliberately informal to enable them to be easier for non-experts to describe process, and for the improvement goals to be indicated on those diagrams. The notation is simple and consists of three elements; Organization Charts, Relationship Maps and Process Maps. Organizational Charts are used to show the reporting dependencies between roles. Relationship Maps are used to indicate dependencies between the nodes of the Organization Chart that are present because of data flows. Process Maps are flowcharts laid out over horizontal roles emphasizing the relationships between the process steps and the organizational elements.

- Managed Process Development Methodology (MPDM)

  MPDM [SPC 90] consists of four related notations and makes heavy use of "template" information. The templates are designed to capture many details of the process that are not present on the diagrams. The four notations of MPDM offer different views of the same process: The Architectural Model - shows composition and specialization of the basic concepts; The Reporting Model - shows which roles report to which; The Interface Model - shows how roles and work products relate to activities; The Behavioral Model - shows the ordering of activities. Incremental steps to complexity: the MPDM approach recognizes that process modeling can be a very complex matter. A single view is not going to capture everything in which you are interested. A graphical notation is not the best for all kinds of information about a process. MPDM allows users to focus on a single viewpoint, draw simple diagrams, and then choose whether to fill in more detail in the related database, or move on to another view of the same process.

- Role Interaction Nets (RIN)

  The RIN formalism [Signh92] is based upon the observation that process descriptions can rarely be derived from a single perspective. Each member of an organization has a view of the process potentially differing from the view of all other members. In order to help reconcile these views, RINs focus on two key themes. The first is the set of organizational role types that communicate and participate in accomplishing goals. The second is the set of interactions that occur between those roles. The Interconnected Roles (IR) model shows how roles are related to one another through relationships such as: binding, assignment, visibility, and interdependencies. Roles are defined as spatially disjoint, asynchronous units that represent self-sufficient contexts for the pursuit of specific organization functions, tasks, or responsibilities. All the process steps that a role is to enact are placed within that role (column). Whenever the process step needs to be carried out in conjunction with other roles, a specific interaction arc appears on the diagram, along with annotations to show the information flow between the roles. While the diagrams are relatively intuitive to understand there is a strong semantic model underlying this method.

- Role Activity Diagrams

  Role Activity Diagrams originated through the study of coordination by Anatol Holt [Hol83]. Its focus is on processes which involve the co-ordination of inter-related activities carried out by people in organizations using a variety of tools. The notation supports four foundation classes: Roles - which represent the individual roles in a process, Actions - individual activities or actions carried out by a role, Entities - data, and structures plus collections of entities and tables of entities, and Interactions - which allow roles to communicate by object passing. Although intuitively simple and easy to read, RADs are a powerful modelling technique which use a number of notational primitives to express very complex process behavior.

- ProNet

  ProNet [Christie 94], a graphical process notation with enactable characteristics, captures a large number of process concepts in a graphical form similar to entity-relationship diagrams, that is a set of entities and a set of relationships are depicted on a single diagram. The notation defines entities as; activities, agents, artifacts, conditions, constraints, and stores and relationships as; has entrance, has exit, instance of, part of, and user defined. Activities form the core of the model. Products and conditions provide the entrance criteria for activity initiation. Activities are responsible for generating exit products and conditions. While the focus of other graphical notations is sometimes to ease human

communication about processes or to emphasize particular aspects of a process, ProNet gives equal status to each of its elements, and focuses on describing a process to the level of detail where an enactable process script could be generated from the diagram.

- Information Mapping™

  This approach (also known as structured writing) comprises a set of strategies for identifying, categorizing and interrelating information. The process of information mapping essentially comprises the systematic presentation of process and reference materials in ordered blocks with marginal labels. However, the arrangement of information in blocks is not simply governed by aesthetic appeal or logic but by the essential contingencies of the defined process. These may include introductions, overviews, purpose, description, activities (with informal diagrams and charts), agents, and artifacts. A further dimension or characteristic of the process of information mapping is its hierarchical organization. This implies that the entire set of information maps and indeed the process of information mapping may proceed from the general to the specific.

- The Multi-View Process modeling project Language (MVP-L)

  MVP-L is a rule-based notation that supports the creation, analysis, and execution of formal process designs [Rombach91]. Its principle use is to develop understanding and communicating process information. A secondary purpose is to guide the execution of processes. MVP-L captures several types of knowledge: project, process, product, attribute, and resource. MVP-L is very readable and easily understood.

References for Appendix A

[SPC 90] Bechtold, Richard, John Brackett, and Sam Redwine, "Process Definition and Modeling Guidebook", SPC-92041-CMC. DTIC AD A279162 (Vol. I) and AD A279199 (Vol. II).

[FIPS 183] "Integration Definition For Function Modeling" (IDEF0), FIPS Pub 183, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, Md. 20899, Issued December 21, 1993

[Reisig 82] Reisig, W., Petri Nets, Published by Springer Verlag, 1982

[Harel 87] Harel, David, "Statecharts: a Visual Formalism for Complex Systems", Science of Computer Programming, 8(3), pp 231-274, 1987.

[RumBrach 90] Rummler, G.A. and A.P. Brache, "Improving Performance - How to Manage the White-Space on the Organization Chart" Jose-Bass Publishers, 1990 ISBN 1-5543-214-4

[Signh 92] Singh, B., and G. L. Rein, "Role Interaction Nets (RINs): A Process Definition Formalism," MCC Technical Report #CT-083-92, July 1992.

[RadPhil 88] Radice, Ronald A., and Richard W. Phillips, "Software Engineering: An Industrial Approach", Prentice-Hall, 1988, ISBN 0-13-823220-2

[Hol83] Holt, A.W., H.R. Ramsey, J.D. Grimes; Coordination System Technology As The Basis For A Programming Environment, Electrical Communication, Volume 57, Number 4, 1983

[Rombach 91] H. Dieter Rombach, "MVP-L, A Language for Process Modeling In-the-Large" University of Maryland Institute for Advanced Computer Studies Technical Report UMIACS-TR-91-96, CS-TR- 2709, Department of Computer Science, University of Maryland, College Park, MD, 20742, 1991.

[Christie 94] Christie, Alan M., "Software Process Automation The Technology and Its Adoption", Springer-Verlag, 1994, ISBN 3-540-58414-5